УДК 681.518.3
ББК 32.973.23
М 54

Автор-составитель А. Н. Семенюта, д-р техн. наук, доцент

Рецензенты: С. Н. Говейко, канд. экон. наук, доцент,
зав. кафедрой коммерческой деятельности
и информационных технологий в экономике
Гомельского государственного университета
им. Ф. Скорины;
С. М. Мовшович, канд. техн. наук, доцент,
зав. кафедрой информационно-вычислительных
систем Белорусского торгово-экономического
университета потребительской кооперации

Рекомендовано к изданию научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 3 от 9 февраля 2010 г.

М 54 **Методы** структурного анализа информационных систем : пособие для студентов специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. А. Н. Семенюта. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2011. – 44 с.
ISBN 978-985-461-833-3

УДК 681.518.3
ББК 32.973.23

В пособии изложены основы методов структурного анализа, которые используются при анализе и проектировании информационных систем.

ISBN 978-985-461-833-3

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Проблема сложности является главной проблемой, которую приходится решать при создании больших и сложных систем любого происхождения, в том числе и информационных систем. Ни один разработчик не в состоянии выйти за пределы человеческих возможностей и понять всю систему в целом. Единственный эффективный подход к решению этой проблемы, который выработало человечество за всю свою историю, заключается в построении сложной системы из небольшого количества крупных частей, каждая из которых, в свою очередь, строится из частей меньшего размера до тех пор, пока самые небольшие части можно будет строить из имеющегося материала.

По отношению к созданию информационных систем это означает, что их необходимо разделять (декомпозировать) на небольшие подсистемы, каждую из которых можно создавать независимо от других. Это позволяет при разработке подсистемы любого уровня держать в уме информацию только о ней, а не обо всех остальных частях системы. Декомпозиция является главным способом преодоления сложности разработки информационных систем.

На сегодняшний день существуют два основных подхода к разработке информационных систем, принципиальное различие между которыми обусловлено разными способами декомпозиции систем. Первый подход называют функционально-модульным или структурным. В его основу положен принцип функциональной декомпозиции, при которой структура системы описывается в терминах иерархии ее функций и передачи информации между отдельными функциональными элементами. Второй подход, объектно-ориентированный, использует объектную декомпозицию. При этом структура системы описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами.

Структурный подход к разработке информационных систем появился в 80-х гг. XX в. и не потерял своей актуальности до настоящего времени. Методы структурного анализа и проектирования стремятся преодолеть сложность информационных систем путем расчленения их на части, т. е. «черные ящики», и иерархической организации этих черных ящиков. Выгода в использовании «черных ящиков» заключается в том, что исследователю или разработчику не всегда требуется знать, как они конкретно устроены, а необходимо знать лишь его входы и выходы, а также его назначение (функцию, которую он выполняет).

При анализе информационных систем наиболее часто применяются следующие методологии структурного анализа, содержащие соответствующие графические и текстовые средства моделирования:

- SADT (Structured Analysis and Design Technique) – метод структурного анализа и проектирования;
- DFD (Data Flow Diagrams) – диаграммы потоков данных совместно со словарями данных и спецификациями процессов;
- ERD (Entity-Relationship Diagrams) – диаграммы «сущность-связь».

Цель данного пособия – ознакомить студентов специальности «Управление информационными ресурсами» с основными понятиями указанных методологий.

В первой части пособия подробно описывается методология структурного анализа и проектирования SADT, приводится пример использования ее для моделирования деятельности организации.

Во второй части подробно описываются технологии диаграмм потоков данных, а также приведены примеры их реального использования.

Третья части пособия посвящена логическому моделированию данных, которые будут храниться и обрабатываться в создаваемой информационной системе.

В целом пособие содержит материал, необходимый студентам для подготовки к экзаменам, практическим занятиям, лабораторным работам, а также для выполнения соответствующих курсовых работ по дисциплинам «Управление жизненным циклом информационных систем» и «Проектирование информационных систем».

1. МЕТОД СТРУКТУРНОГО АНАЛИЗА И ПРОЕКТИРОВАНИЯ SADT

1.1. Общие сведения

Метод SADT представляет собой совокупность правил и процедур, предназначенных для построения функциональной модели исследуемой системы.

Функциональная направленность означает, что функции системы исследуются независимо от механизмов, которые обеспечивают их выполнение. «Функциональная» точка зрения позволяет четко отделить аспекты назначения системы от аспектов ее физической реализации. В целом метод SADT направлен на изучение того, что делает исследуемая система, а не каким конкретно способом.

Метод SADT успешно используется в военных, промышленных и коммерческих организациях США для решения широкого круга задач, таких, как долгосрочное и стратегическое планирование, автоматизированное производство и проектирование, разработка программного обеспечения, управление финансами и материально-техническим снабжением и др. Метод SADT поддерживается Министерством обороны США, которое было инициатором разработки стандарта IDEF0 (Icam DEFinition) – подмножества SADT, являющегося основной частью программы ICAM (Integrated Computer Aided Manufacturing – интегрированная компьютеризация производства). Позднее IDEF0 был утвержден в качестве федерального стандарта США.

Метод SADT может использоваться для моделирования самых разнообразных систем. При анализе существующих систем метод SADT может применяться для выявления функций, выполняемых системой, и указания механизмов, посредством которых они осуществляются.

1.2. Основные элементы IDEF0-диаграмм

Основным элементом IDEF0-диаграмм является функциональный блок с соответствующими входящими и исходящими стрелками (рисунок 1).

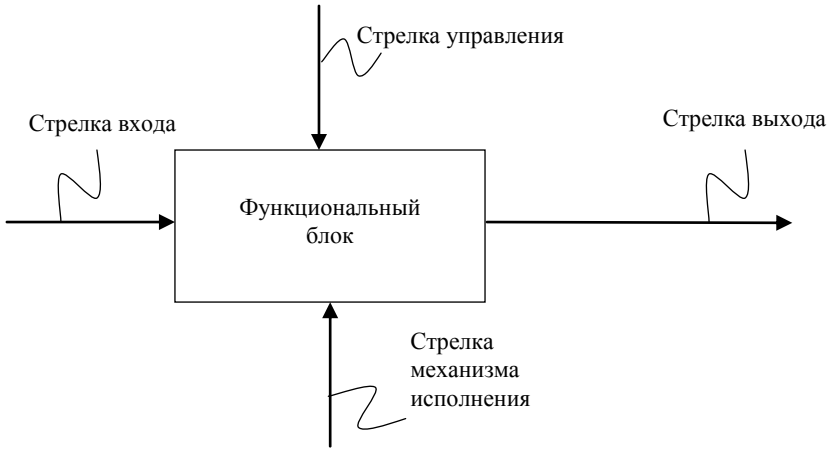

Рисунок 1 – **Функциональный блок IDEF0-диаграмм**

Графически функциональный блок изображается в виде прямоугольника и моделирует некоторую конкретную функцию в рамках рассматриваемой системы.

Название каждого функционального блока должно быть образовано с использованием глаголов или отглагольных существительных. Важно подбирать имена функциональных блоков так, чтобы они отражали точку зрения, используемую для моделирования.

Примеры наименований функциональных блоков:

- Вычисление суммы к оплате;
- Расчет подоходного налога;
- Систематизация информации.

На IDEF0-диаграммах возможны четыре типа стрелок (см. рисунок 1):

- *стрелка входа* – отображает то, что подается на вход функционального блока;
- *стрелка управления* – отображает ограничения и инструкции, влияющие на работу функционального блока;
- *стрелка выхода* – отображает то, что является результатом работы функционального блока;
- *стрелка механизма исполнения* – отображает то, что используется для работы функционального блока.

Стрелки входа. Вход представляет собой сырье или информацию, потребляемую или преобразуемую функциональным блоком для производства выхода. Стрелки входа всегда направлены в левую сторону

прямоугольника, обозначающего в IDEF0 функциональный блок. Наличие входных стрелок на диаграмме не является обязательным, так как возможно, что некоторые блоки ничего не преобразуют и не изменяют. Примером блока, не имеющего входа, может служить блок «Принятие решения руководством», где анализируется несколько факторов, но ни один из них непосредственно не преобразуется и не потребляется в результате принятия какого-либо решения.

Стрелки управления. Стрелки управления отвечают за регулирование того, как и когда выполняется функциональный блок. Так как управление контролирует поведение функционального блока для обеспечения создания желаемого выхода, каждый функциональный блок должен иметь как минимум одну стрелку управления. Стрелки управления всегда входят в функциональный блок сверху.

Управление часто существует в виде правил, инструкций, законов, политики, набора необходимых процедур или стандартов. Влияя на работу блока, оно само остается неизменным. Может оказаться, что целью функционального блока является как раз изменение того или иного правила, инструкции, стандарта и т. п. В этом случае стрелка, содержащая соответствующую информацию, должна рассматриваться не как управление, а как вход функционального блока.

Управление можно рассматривать как специфический вид входа. В случаях, когда неясно, относить ли стрелку к входу или к управлению, предпочтительно относить ее к управлению до момента, пока неясность не будет разрешена.

Стрелки выхода. Выход – это продукция или информация, получаемая в результате работы функционального блока. Каждый блок должен иметь как минимум один выход. Действие, которое не имеет никакого четко определяемого выхода, желательно не моделировать вообще.

Стрелки механизма исполнения. Механизм является ресурсом, который непосредственно исполняет моделируемое действие. С помощью механизмов исполнения могут моделироваться ключевой персонал, техника и (или) оборудование. Стрелки механизма исполнения могут отсутствовать в случае, если оказывается, что они не являются необходимыми для достижения поставленной цели моделирования.

Входящие и исходящие стрелки функциональных блоков отображают элементы исследуемой системы, которые обрабатываются функциональным блоком или оказывают иное влияние на функцию, отображенную данным функциональным блоком. С помощью стрелок отображают различные объекты, в той или иной степени определяю-

щие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т. д.) или потоки данных и информации (документы, данные, инструкции и т. д.).

Каждая стрелка должна иметь свое уникальное наименование. Для названий стрелок, как правило, употребляются имена существительные.

В качестве примера на рисунке 2 приведен функциональный блок IDEF0-диаграммы, моделирующий обработку отчетности налоговым инспектором.

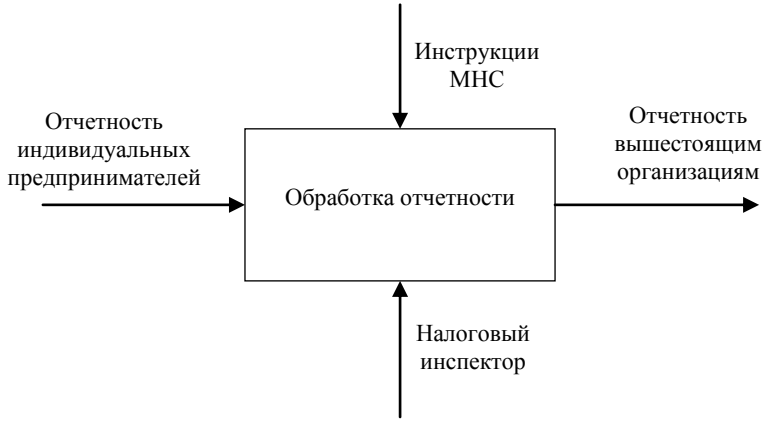

Рисунок 2 – **Пример IDEF0-диаграммы**

Из приведенного на рисунке 2 примера следует, что налоговый инспектор на основании ведомственных инструкций из отчетности, предоставляемой индивидуальными предпринимателями, формирует отчетность для вышестоящих организаций. При этом в явном виде алгоритм преобразования отчетности индивидуальных предпринимателей в отчетность для вышестоящих организаций не указывается. В этом и состоит суть IDEF0-диаграмм – отражать, что делается, а не как.

На одной IDEF0-диаграмме может быть несколько функциональных блоков, при этом выход одного функционального блока может использоваться в нескольких других блоках.

На рисунке 3 приведен фрагмент IDEF0-диаграммы, когда один из блоков должен полностью завершить работу перед началом работы другого блока (прием заказа должен предшествовать выписке счета).

Рисунок 3 – **Пример IDEF0-диаграммы**

На рисунке 4 приведен фрагмент IDEF0-диаграммы, когда один блок управляет работой другого (принципы формирования инвестиционного портфеля влияют на поведение брокеров на бирже).

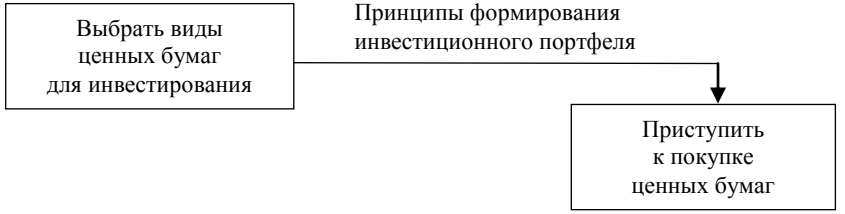

Рисунок 4 – **Пример IDEF0-диаграммы**

На рисунке 5 приведен фрагмент IDEF0-диаграммы, когда выход одного функционального блока является механизмом исполнения другого блока (зажим, используемый для закрепления детали, должен быть собран для того, чтобы выполнить сборку детали).

Рисунок 5 – **Пример IDEF0-диаграммы**

Обратные связи на вход и на управление применяются в случаях, когда зависимые блоки формируют обратные связи для управляющих ими блоков (например, получаемая от брокеров информация о текущих биржевых курсах применяется для корректировки стратегии игры на бирже) (рисунок 6).

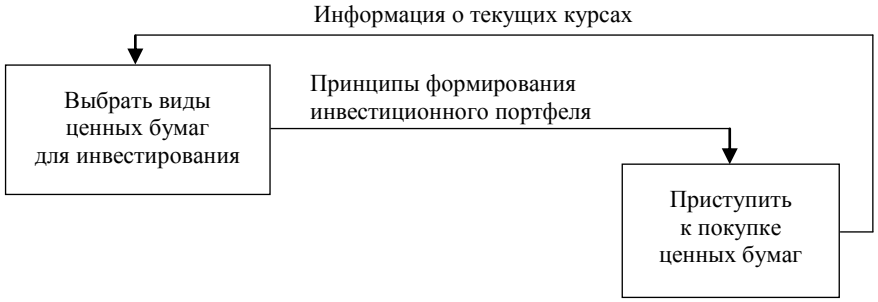

Рисунок 6 – **Пример IDEF0-диаграммы**

При построении IDEF0-диаграмм возможно как разъединение, так и соединение стрелок на диаграмме. Разъединенные на несколько частей стрелки могут иметь наименования, отличающиеся от наименования исходной стрелки. Исходная и разъединенные (или объединенные) стрелки в совокупности называются связанными. Такая техника обычно применяется для того, чтобы отразить использование в процессе только части сырья или информации, обозначаемой исходной стрелкой (рисунок 7). Аналогичный подход применяется по отношению к объединенным стрелкам.

Рисунок 7 – **Пример IDEF0-диаграммы**

1.3. Построение иерархии диаграмм

Построение IDEF0-диаграмм всегда начинается с представления исследуемой системы как единого целого – одного функционального блока с входами и выходами, направленными за пределы изучаемой

системы. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой и обозначается идентификатором «A-0».

В пояснительном тексте к контекстной диаграмме должна быть указана цель построения диаграммы (в виде краткого описания) и зафиксирована точка зрения. Определение и формализация цели разработки IDEF0-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Например, функциональные модели одной и той же организации, с точки зрения главного технолога и финансового директора, будут существенно различаться по направленности их детализации.

Далее производится декомпозиция построенной контекстной диаграммы. Декомпозиция позволяет постепенно и структурированно представить модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

В процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы, и называется дочерней по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется дочерним блоком). В свою очередь, функциональный блок-предок называется родительским блоком по отношению к дочерней диаграмме, а диаграмма, к которой он принадлежит, – родительской диаграммой.

Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. Важно отметить, что в каждом случае декомпозиции функционального блока все стрелки, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0-модели. Наглядно принцип декомпозиции представлен на рисунке 8.

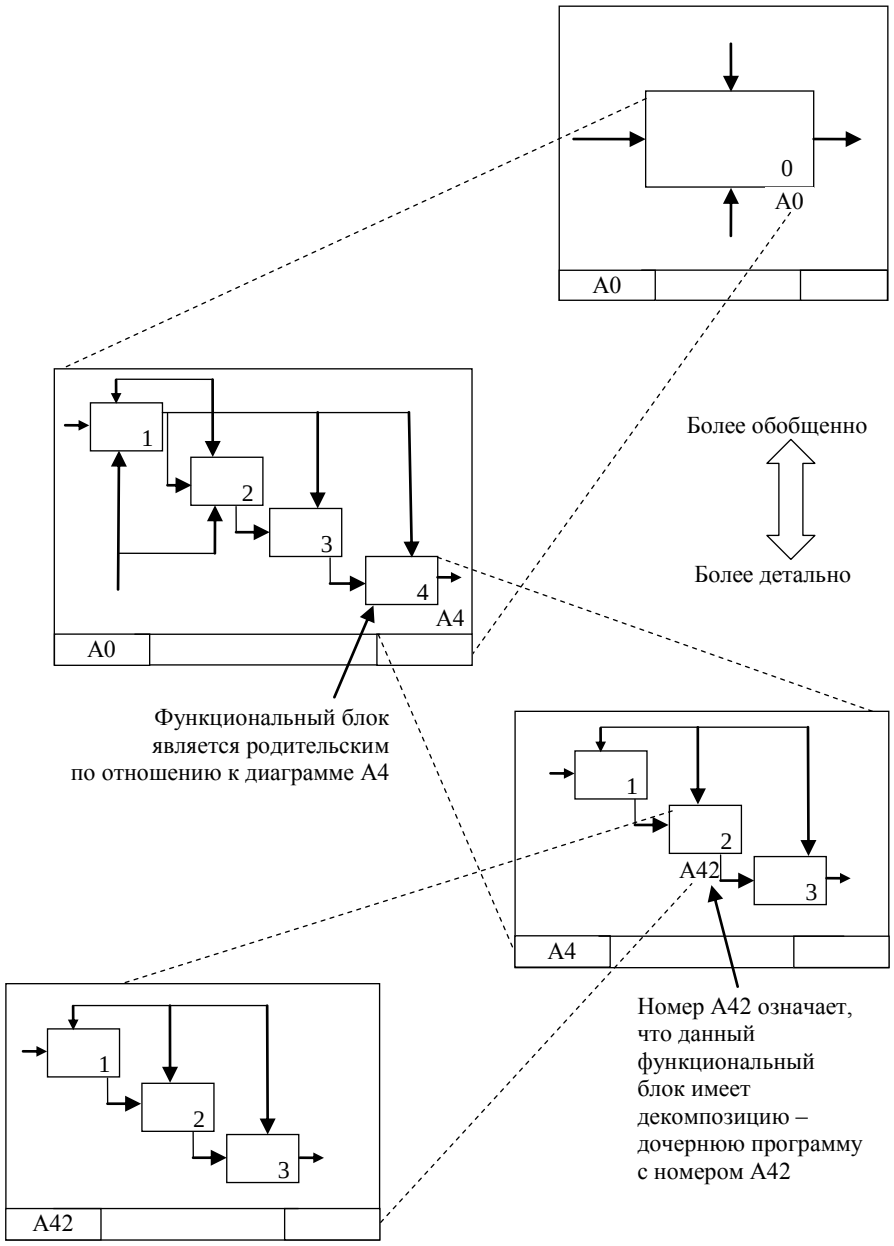
0
A0
A0
1
2
3
4
A4
A0
Более обобщенно
Более детально
Функциональный блок
является родительским
по отношению к диаграмме А4
1
2
A42
3
A4
Номер А42 означает,
что данный
функциональный
блок имеет
декомпозицию –
дочернюю программу
с номером А42
1
2
3
A42

Рисунок 8 – **Декомпозиция функциональных блоков IDEF0-диаграммы**

Часто бывают случаи, когда отдельные стрелки (входы-выходы) не имеет смысла продолжать рассматривать в дочерних диаграммах ниже какого-то определенного уровня в иерархии, или наоборот – они не имеют практического смысла выше какого-то уровня.

Например, стрелку, моделирующую «деталь» на входе в функциональный блок «Обработать на токарном станке», не имеет смысла отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. С другой стороны, возникает необходимость избавиться от отдельных «концептуальных» стрелок и не детализировать их глубже некоторого уровня.

Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования. Изображение «туннеля» в виде двух квадратных скобок вокруг начала стрелки означает, что эта стрелка не была унаследована от функционального родительского блока и появилась (из «туннеля») только на этой диаграмме. В свою очередь, такое же изображение вокруг конца стрелки в непосредственной близи от блока-приемника означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта стрелка отображаться и рассматриваться не будет.

Следует обратить внимание на взаимосвязь нумерации функциональных блоков и диаграмм – каждый блок имеет свой уникальный порядковый номер на диаграмме (цифра в правом нижнем углу прямоугольника), а под правым углом указывают номер дочерней для этого блока диаграммы. Отсутствие этого обозначения говорит о том, что декомпозиции для данного блока не существует.

Для однозначного толкования названий стрелок и функциональных блоков, встречающихся на IDEF0-диаграммах, часто разрабатывают так называемый глоссарий (Glossary). В нем для каждого из элементов IDEF0 – диаграмм, функциональных блоков, интерфейсных дуг – создаются соответствующие определения, которые характеризуют объект, отображенный данным элементом. Например, для стрелки с названием «распоряжение об оплате» глоссарий может содержать перечень полей соответствующего документа, необходимый набор виз и т. д. Глоссарий гармонично дополняет диаграммы необходимой информацией.

1.4. Пример построения IDEF0-диаграмм деятельности организации

1.4.1. Постановка задачи

Небольшая организация – открытое акционерное общество (ООО) «Компьютер» – занимается сборкой, а затем оптовой продажей компьютеров. Для этого она закупает комплектующие для сборки компьютеров от трех поставщиков, собирает из них настольные компьютеры и ноутбуки, а затем продает их корпоративным клиентам.

За последние три месяца показатели прибыльности ООО «Компьютер» значительно ухудшились, что в перспективе может привести к неудовлетворительным финансовым результатам за год. Директору организации необходимо предпринимать какие-то действия, поэтому он решает провести анализ деятельности организации с использованием SADT-метода.

1.4.2. Построение контекстной диаграммы

Анализ деятельности любого объекта исследования с использованием SADT-метода всегда начинается с построения контекстной диаграммы (диаграммы нулевого уровня), которая изображает исследуемый объект как единое целое со всеми связями с внешней средой.

Основная функция ООО «Компьютер», с точки зрения директора, – это продажа компьютеров, собранных из закупаемых комплектующих, поэтому контекстная диаграмма будет иметь вид, соответствующий рисунку 9.

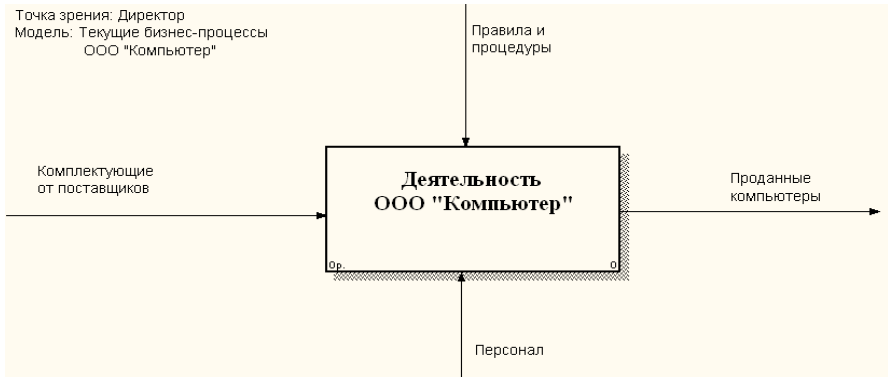

Рисунок 9 – **Контекстная диаграмма деятельности ООО «Компьютер»**

1.4.3. Построение диаграммы декомпозиции первого уровня

Существующая технология работы ООО «Компьютер» предполагает следующее:

- менеджер по продажам обзванивает потенциальных клиентов, принимает от них заказы, а затем передает их в сборочный цех для выполнения;
- техники из получаемых со склада комплектующих собирают компьютеры;
- собранные компьютеры передаются на склад для отгрузки клиентам;
- неисправные комплектующие возвращаются на склад;
- работник склада отгружает клиентам собранные компьютеры, принимает присылаемые от поставщиков комплектующие, а также отсылает поставщикам неисправные комплектующие.

Таким образом, основными функциями, реализуемыми ООО «Компьютер», являются:

- прием заказов;
- сборка компьютеров;
- отгрузка заказов;
- прием и возврат комплектующих.

Полученная информация позволяет построить диаграмму декомпозиции первого уровня деятельности ООО «Компьютер» (рисунок 10).

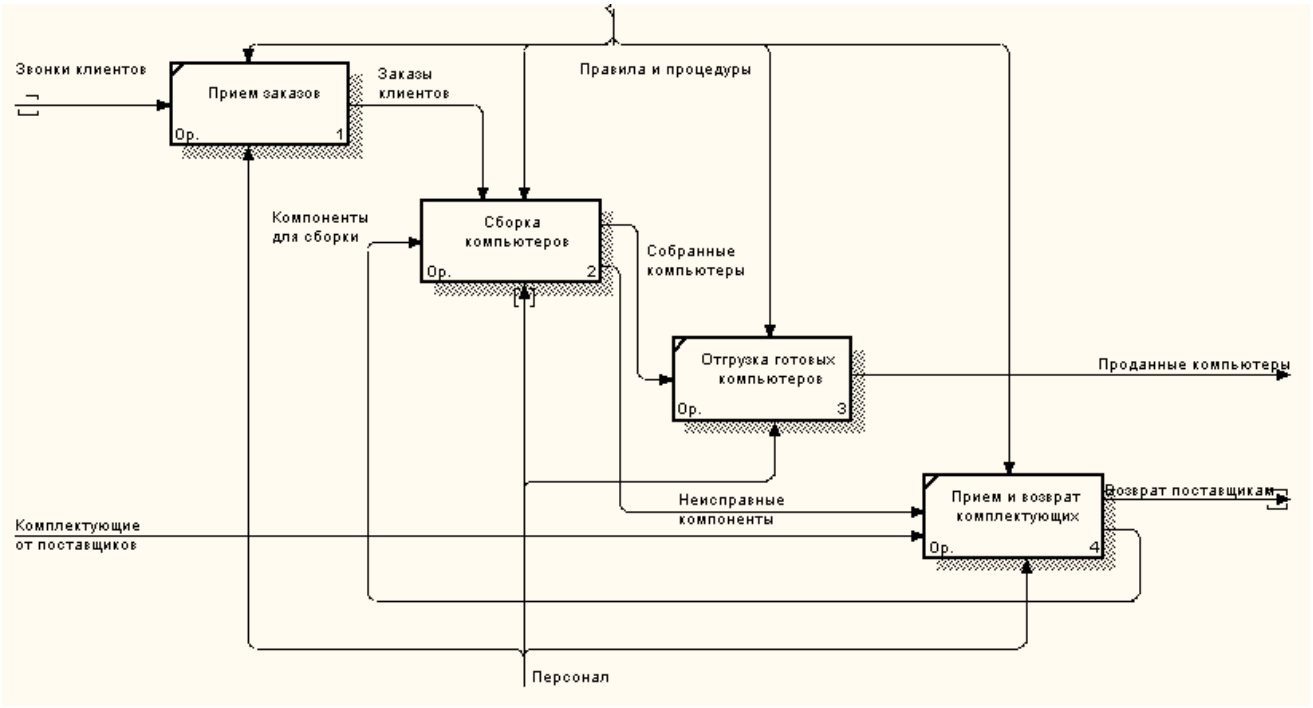

Рисунок 10 – **Декомпозиции контекстной диаграммы**

1.4.4. Построение диаграммы декомпозиции функционального блока «Сборка компьютеров»

Существующая технология работы ООО «Компьютер» предполагает следующее:

- производственный отдел получает заказы клиентов из отдела продаж по мере их поступления;
- диспетчер координирует работу сборщиков, сортирует заказы, группирует их и дает указание на отгрузку компьютеров, когда они готовы;
- диспетчер группирует заказы отдельно для настольных компьютеров и ноутбуков и направляет на участок сборки;
- техники-сборщики собирают компьютеры согласно спецификациям заказа и инструкциям по сборке;
- техник-тестировщик тестирует каждый компьютер и в случае необходимости заменяет неисправные компоненты;
- техник-тестировщик направляет результаты тестирования диспетчеру, который на основании этой информации принимает решение о передаче компьютеров из соответствующей группы заказов на отгрузку.

На основе этой информации произведена декомпозиция функционального блока «Сборка компьютеров» на четыре функциональных блока так, как показано на рисунке 11.

1.4.5. Результаты анализа

Полученная в результате проведения экспертизы информация, а также диаграммы декомпозиции явно указывают, что в исследуемой организации в явном виде отсутствует функция, связанная с систематической работой с клиентами (маркетинг). Это может быть одной из причин снижения прибыльности.

В свою очередь, из диаграммы декомпозиции функционального блока «Сборка компьютеров» видно, что тестированием собранных компьютеров занимаются техники-тестировщики. Однако эту работу вполне можно поручить техникам-сборщикам с сокращением должности техника-тестировщика.

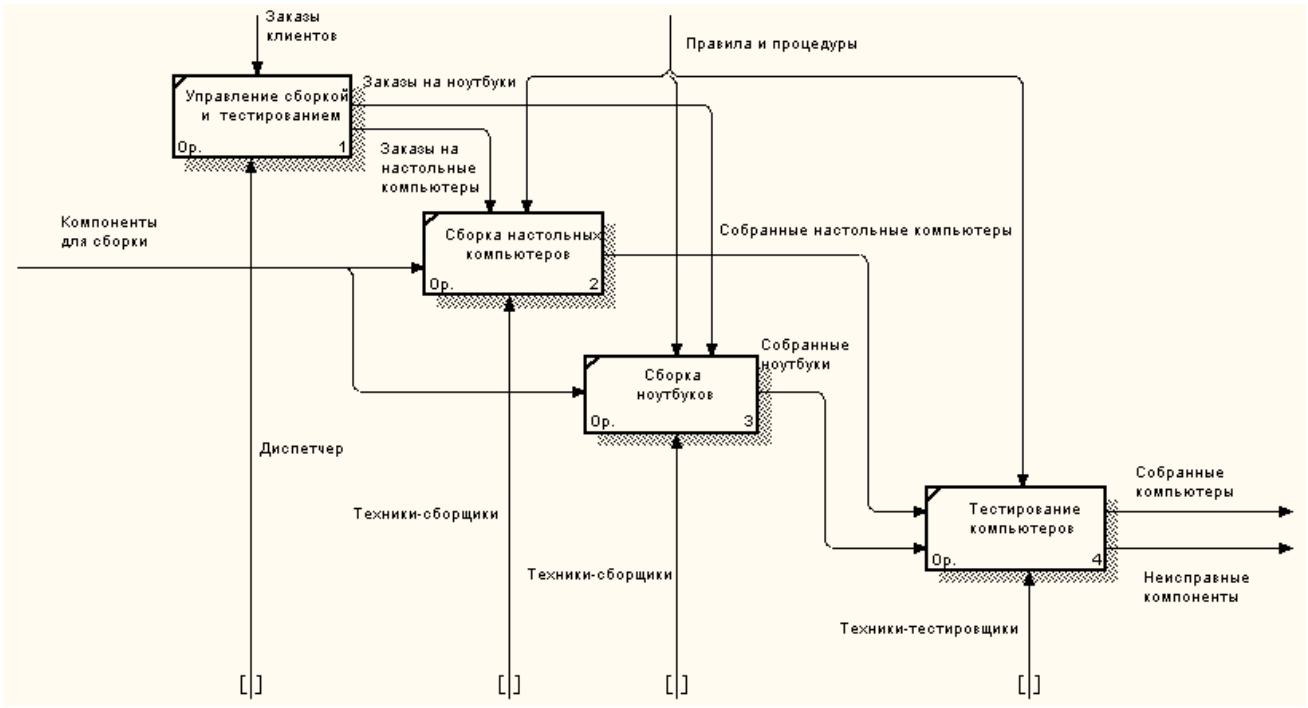

Рисунок 11 – **Диаграмма декомпозиции функционального блока «Сборка компьютеров»**

2. ДИАГРАММЫ ПОТОКОВ ДАННЫХ

2.1. Общие сведения

Идея, лежащая в основе диаграмм потоков данных (DFD), родилась в 20-х гг. XX в. Именно тогда осуществлявший реорганизацию переполненного клерками офиса консультант обозначил кружком каждого клерка, а стрелкой – каждый документ, передаваемый между ними. Используя такую диаграмму, он предложил схему реорганизации, в соответствии с которой двое клерков, обменивающиеся множеством документов, были посажены рядом, а клерки с малым взаимодействием были посажены на большом расстоянии. Так появилась первая модель, предназначенная для анализа потоков данных в организации.

Современные диаграммы потоков данных в основном применяются для моделирования реально циркулирующей внутри исследуемой системы информации, хотя в принципе они могут быть использованы для анализа перемещений любых материальных объектов, а не только для моделирования потоков информации. При этом исследуемая система представляется в виде сети процессов, связанных потоками данных. Главная цель такого представления – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

2.2. Основные элементы DFD-диаграмм

Для изображения диаграмм потоков данных традиционно используются нотации Йордана (Yourdon) и Гейна-Сарсона (Gane and Sarson), отличие между которыми заключается в различных условных обозначениях для основных элементов диаграмм потоков данных (рисунок 12).

Потоки данных на диаграммах обычно изображаются именованными стрелками, ориентация которых указывает направление движения информации.

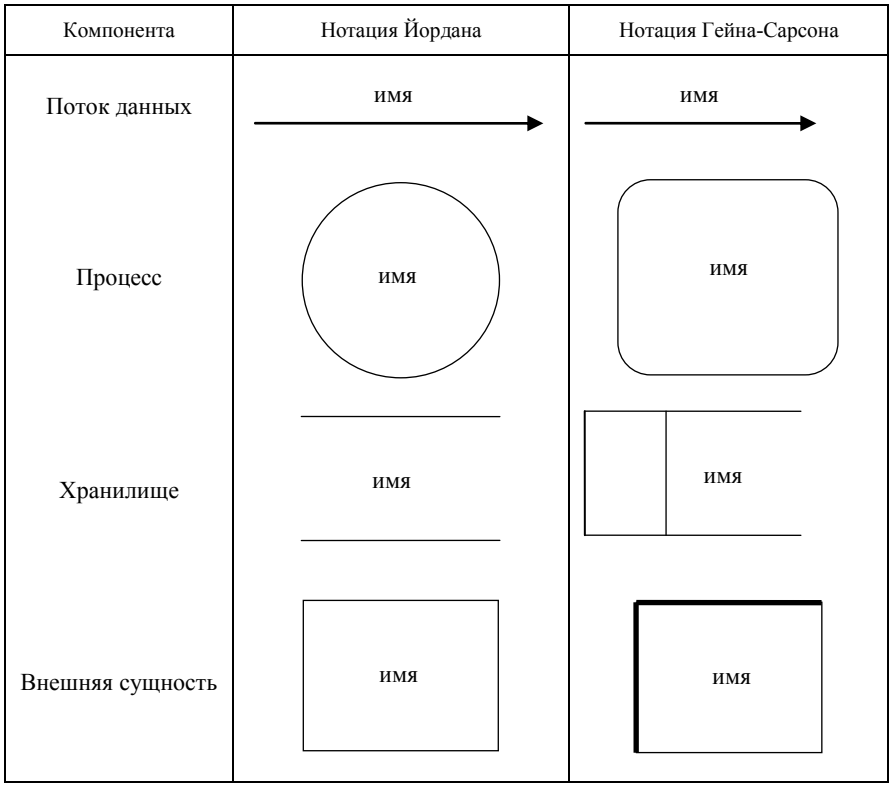

Рисунок 12 – **Основные элементы диаграмм потоков данных**

Иногда информация передается в одном направлении, обрабатывается, а затем возвращается назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним – двунаправленным.

Назначение *процесса* состоит в преобразовании того, что поступает на его вход, в то, что появляется на его выходе в соответствии с действием, задаваемым именем процесса. Это имя должно быть выражено глаголом в неопределенной форме с последующим дополнением (например, Обработать кредитную карту) либо отглагольным существительным (например, Обработка кредитной карты).

Кроме того, каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Наиболее часто процессы на DFD-диаграммах используются для моделирования преобразований информации, поступающей на его вход, в информацию, выдаваемую на его выход.

В *хранилища данных* можно временно поместить информацию, которая перемещается между процессами. Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Имя хранилища должно идентифицировать его содержимое и быть выражено существительным.

Внешняя сущность может быть любым объектом за пределами исследуемой системы, являющимся источником или приемником информации из исследуемой системы. Ее имя должно быть выражено существительным, например, Пользователь.

На рисунке 13 в качестве примера приведен фрагмент DFD-диаграммы, моделирующей прием заказов по телефону в магазине, работающем по принципу доставки заказов на дом.

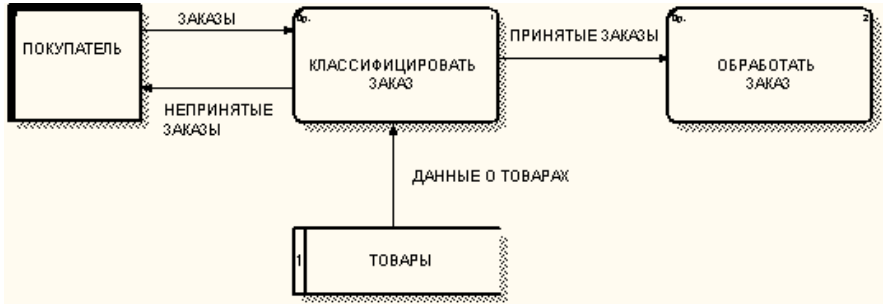

Рисунок 13 – **Пример DFD-диаграммы**

2.3. Построение иерархии DFD-диаграмм

Построение DFD-диаграмм исследуемой системы, также как и IDEF0-диаграмм, всегда начинается с построения контекстной диаграммы.

Контекстная DFD-диаграмма обычно состоит из одного функционального блока со стрелками, соединенными с внешними сущностями. Функциональный блок на этой диаграмме обычно имеет имя, совпадающее с именем исследуемой системы.

Пример контекстной диаграммы приведен на рисунке 14.

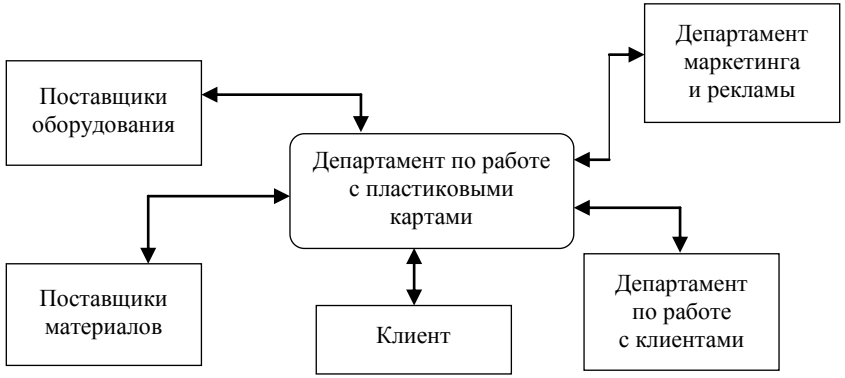

Рисунок 14 – **Пример контекстной DFD-диаграммы**

Контекстная диаграмма четко фиксирует исследуемую систему вместе с информационными потоками, связывающими ее с внешним миром. Она идентифицирует источники или приемники информации из исследуемой системы (внешние сущности), а также единственный процесс, отражающий главную цель или природу исследуемой системы насколько это возможно. И хотя контекстная диаграмма выглядит тривиальной, несомненная ее полезность заключается в том, что она устанавливает границы анализируемой системы. Каждый проект должен иметь ровно одну контекстную диаграмму, при этом нет необходимости в нумерации единственного ее процесса.

Далее в процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма содержит функциональные блоки, отображающие главные процессы контекстной диаграммы, и называется дочерней по отношению к нему. Аналогичным образом каждый из процессов дочерней диаграммы может быть детализирован далее. Такая процедура последовательной декомпозиции производится до тех пор, пока не будет получен комплект диаграмм, удовлетворяющих исследователя.

При таком построении иерархии DFD каждый процесс более низкого уровня необходимо соотнести с процессом верхнего уровня. Обычно для этой цели используются структурированные номера процессов. Так, например, при детализации процесса номер 2 на диаграмме первого уровня (раскрывая его с помощью DFD), содержащей три процесса, их номера будут иметь следующий вид: 2.1, 2.2 и 2.3. При необходимости можно перейти на следующий уровень, т. е. для процесса 2.2 получим 2.2.1, 2.2.2. и т. д.

2.4. Пример построения DFD-диаграмм

Рассмотрим последовательность построения DFD-диаграмм, моделирующих работу банкомата по обслуживанию клиента по его кредитной карте.

Проведя первоначальную экспертизу, можно выявить, что для банковского обслуживания клиенту необходимо вставить в банкомат свою кредитную карту для автоматического считывания с нее служебной информации (пароль, лимит денег, детали клиента), а также сообщить свои ключевые данные, а именно пароль и запрос на обслуживание, т. е. требуемую ему услугу (например, снятие со счета наличных денег).

В свою очередь, банкомат должен обеспечить следующее:

- выдать сообщение, приглашающее клиента ввести ключевые данные;
- передать в компьютер банка протокол обслуживания, в котором содержится информация об обработанной документации, изымаемой денежной сумме и данные по истории запроса;
- получить от компьютера банка данные по счету клиента в банке;
- выдать клиенту деньги;
- выдать клиенту выписку по проведенному обслуживанию, включающую выписку о деньгах, выписку по балансу и выписку по операции, проведенной банком.

Полученная информация позволяет построить контекстную диаграмму (рисунок 15).

Рисунок 15 – **Контекстная диаграмма**

Далее данная контекстная диаграмма может быть декомпозирована, как показано на рисунке 16. Эта диаграмма содержит четыре процесса и одно хранилище данных.

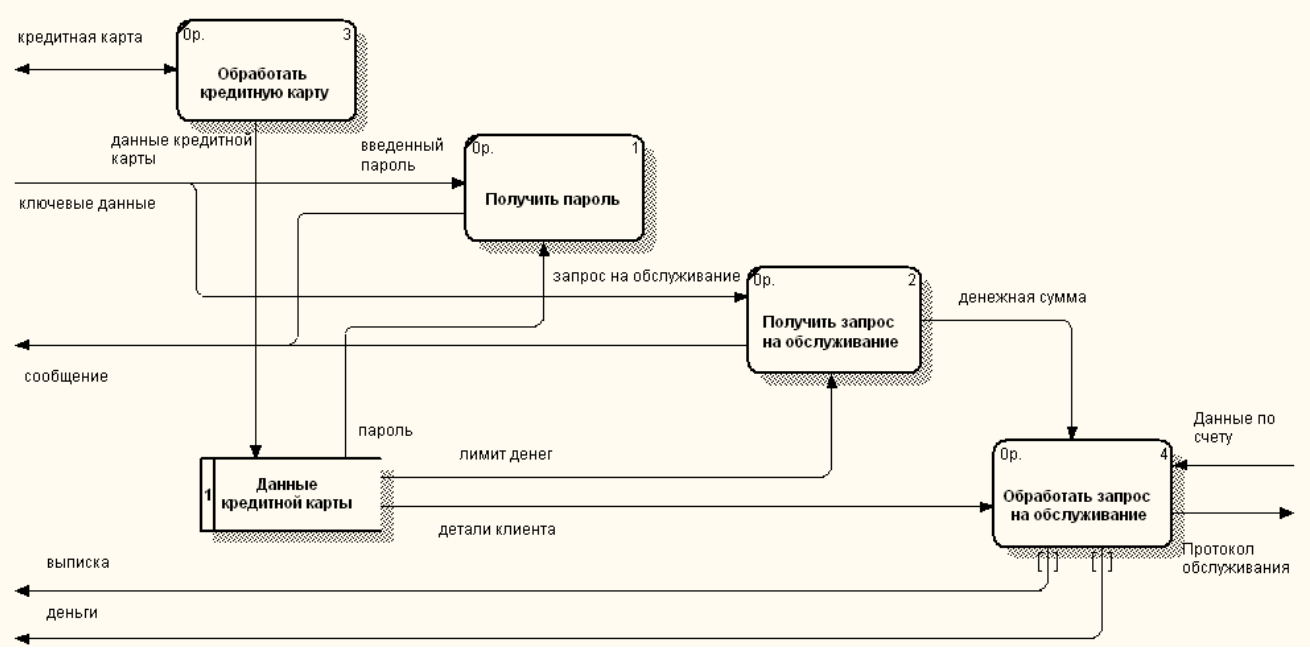

Рисунок 16 – **Диаграмма декомпозиции первого уровня**

Процесс Получить пароль осуществляет прием и проверку пароля клиента и имеет на входе (выходе) следующие потоки:

- внешний выходной поток сообщение для информирования клиента о своей готовности принять пароль;
- входной поток введенный пароль как элемент внешнего потока ключевые данные;
- входной поток пароль из хранилища Данные кредитной карты для проверки вводимого клиентом пароля.

Процесс Получить запрос на обслуживание осуществляет прием и проверку запроса клиента на проведение необходимой ему банковской операции и имеет на входе (выходе) следующие потоки:

- внешний выходной поток сообщение для информирования клиента о своей готовности принять запрос на обслуживание;
- входной поток запрос на обслуживание как элемент внешнего потока ключевые данные;
- входной поток лимит денег из хранилища Данные кредитной карты для контроля наличия денег на счете клиента.

Процесс Обработать запрос на обслуживание имеет внешний входной поток данные по счету (из внешней сущности Компьютер банка), входной поток детали клиента (из хранилища), а также внешние выходные потоки выписка, деньги и протокол обслуживания.

Процесс Обработать кредитную карту осуществляет считывание информации с кредитной карты и имеет на входе внешний поток кредитная карта, а на выходе поток данные кредитной карты.

Процессы Получить пароль, Получить запрос на обслуживание и Обработать кредитную карту являются элементарными, поэтому нет необходимости в их дальнейшей детализации. Процесс Обработать запрос на обслуживание может быть детализирован с помощью DFD второго уровня (рисунок 17). Эта диаграмма содержит четыре элементарных процесса.

Процесс Обработать документацию банка осуществляет обработку внутренней банковской документации по клиенту и имеет входной поток детали клиента и выходной поток обработанная документация (часть внешнего потока протокол обслуживания).

Процесс Распечатать баланс клиента выдает справку по истории счета клиента и по балансу клиента. Входные потоки – детали клиента и данные по балансу (часть внешнего потока данные по счету), выходные потоки – выписка по балансу (часть внешнего потока выписка) и данные по истории запроса (часть внешнего потока протокол обслуживания).

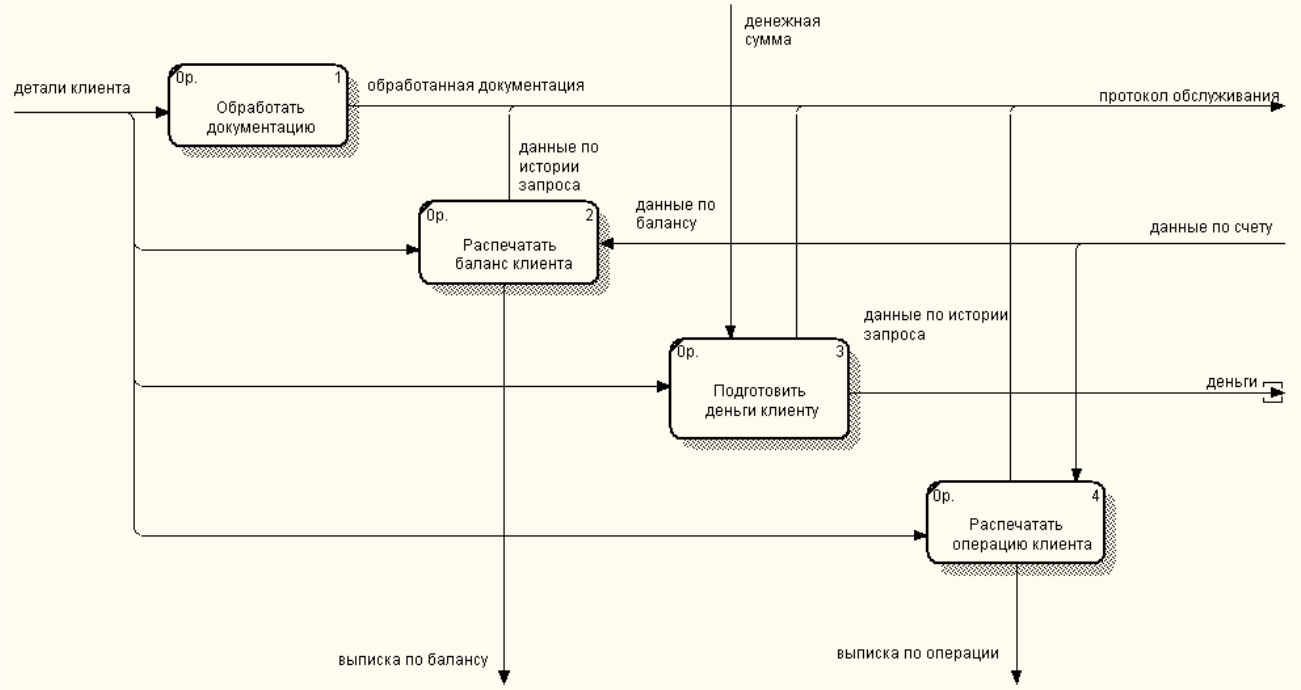

Рисунок 17 – **Детализация процесса Обработать запрос на обслуживание**

Процесс Подготовить деньги клиенту обеспечивает выдачу наличных денег и информирование компьютера банка об изъятых из банка деньгах. Он имеет входные потоки денежная сумма и детали клиента, и выходные потоки деньги и денежная сумма (часть потока протокол обслуживания).

Процесс Распечатать операцию клиента выдает справку по истории счета и уведомление по проведенной операции. Входные потоки – данные по счету и детали клиента, выходные потоки – выписка по операции (часть потока выписка) и данные по истории запроса (часть потока протокол обслуживания).

2.5. Словарь данных

Диаграммы потоков данных не отражают в деталях содержимое циркулирующей между процессами информации. В то же время при анализе и проектировании информационных систем часто необходимо иметь точные описания потоков данных.

Для решения этой задачи обычно DFD-диаграммы дополняются так называемыми словарями данных. В словаре данных хранится детальная информация о структуре и отдельных элементах этой структуры для каждого потока в исследуемой системе.

Простейшим способом описания структуры потока данных является использование алгебраической нотации. В данной нотации используются следующие символы:

- знак «=» означает «состоит из»;
- знак «+» означает «и»;
- квадратные скобки моделируют ситуацию, когда элемент данных может принимать одно и только одно значение из списка в скобках;
- круглые скобки моделируют ситуацию, когда элемент данных не обязательно может присутствовать в потоке.

Примеры записей в словаре данных

НОМЕР ТЕЛЕФОНА = КОД СЕТИ + НОМЕР В СЕТИ

Эта запись означает, что поток данных НОМЕР ТЕЛЕФОНА состоит из двух элементов – КОД СЕТИ и НОМЕР В СЕТИ.

ТИП КРЕДИТНОЙ КАРТЫ = [VISA | MASTERCARD]

Данная запись означает, что поток данных ТИП КРЕДИТНОЙ КАРТЫ состоит из элемента, принимающего одно из двух взаимоисключающих значений.

ФИО = ИМЯ + ФАМИЛИЯ + (ОТЧЕСТВО)

Запись означает, что хотя поток данных ФИО состоит из трех элементов, реально последнего элемента в нем может не быть, и тогда на его месте передаются заранее установленные символы (например, пробелы).

Для описания отдельных элементов данных удобно применять стандартные способы задания типов данных, используемые в языках программирования:

ИМЯ = CHAR (20)

Эта запись означает, что данный элемент состоит максимум из двадцати символов.

2.6. Спецификации процессов DFD-диаграмм

Диаграммы потоков данных также не отражают в деталях, каким конкретно образом входная информация преобразуется в выходную каждым имеющимся процессом. Однако, если на основании DFD-диаграмм будет создаваться реальная информационная система, то без описания четких алгоритмов функционирования каждого процесса не обойтись. Поэтому для решения этой задачи DFD-диаграммы должны быть дополнены так называемыми спецификациями процессов.

Известно большое число разнообразных методов, позволяющих описать алгоритмы переработки информации соответствующими процессами.

Одним из вариантов читабельного, строгого описания спецификаций процессов DFD-диаграмм является использование структурированного естественного языка. Он является разумной комбинацией строгости языка программирования и читабельности естественного языка и состоит из подмножества слов, организованных в определенные логические структуры, арифметических выражений и диаграмм.

В состав языка входят следующие основные символы:

- глаголы, ориентированные на действие и применяемые к объектам;

• термины, определенные на любой стадии проекта создания информационной системы (например, задачи, процедуры, символы данных и т. п.);
• предлоги и союзы, используемые в логических отношениях;
• общеупотребительные математические, физические и технические термины;
• арифметические уравнения;
• таблицы, диаграммы, графы;
• комментарии.

При использовании структурированного естественного языка спецификации процессов строятся на основе типовых конструкций структурного программирования:

1. Последовательность

ВЫПОЛНИТЬ функция 1
ВЫПОЛНИТЬ функция 2
ВЫПОЛНИТЬ функция 3

2. Выбор

ЕСЛИ <условие> **ТО**
    ВЫПОЛНИТЬ функция 1
ИНАЧЕ
    ВЫПОЛНИТЬ функция 2
КОНЕЦ ЕСЛИ

3. Итерация

ДЛЯ <условие>
    ВЫПОЛНИТЬ функция
КОНЕЦ ДЛЯ

или

ДО ТЕХ ПОР ПОКА <условие>
    ВЫПОЛНИТЬ функция
КОНЕЦ ДО ТЕХ ПОР ПОКА

При использовании структурированного естественного языка приняты следующие соглашения:

• Логика процесса выражается в виде комбинации последовательных конструкций, конструкций выбора и итераций.

• Ключевые слова ЕСЛИ, ВЫПОЛНИТЬ, ИНАЧЕ должны быть написаны прописными буквами.

• Слова или фразы, определенные в словаре данных, также должны быть написаны прописными буквами.

• Глаголы должны быть активными, недвусмысленными и ориентированными на целевое действие (заполнить, вычислить, извлечь, а не модернизировать, обработать).

• Логика процесса должна быть выражена четко и недвусмысленно.

Например, спецификация процесса 1.1 (Получить пароль) для диаграммы, изображенной на рисунке 16, с использованием структурированного естественного языка будет иметь следующий вид:

ДО ТЕХ ПОР ПОКА
введенный пароль ≠ пароль или были сделаны три попытки ввода
ВЫПОЛНИТЬ выдать СООБЩЕНИЕ клиенту, запрашивающее ввод пароля
ВЫПОЛНИТЬ принять ВВЕДЕННЫЙ ПАРОЛЬ
ВЫПОЛНИТЬ установить флаг КОРРЕКТНЫЙ ПАРОЛЬ в случае равенства введенного пароля паролю для карточки
КОНЕЦ ДО ТЕХ ПОР ПОКА

3. ДИАГРАММЫ «СУЩНОСТЬ-СВЯЗЬ»

3.1. Общие сведения

Диаграммы «сущность-связь» (ERD) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними. Фактически с помощью ERD осуществляется детализация хранилищ данных проектируемой системы, а также документируются сущности системы и способы их взаимодействия, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее общими понятиями, использующимися при построении любых диаграмм «сущность-связь», являются:

• сущность;
• связь;
• атрибут.

Сущность (*Entity*) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области.

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами, а именно:

• иметь уникальное имя, к одному и тому же имени должна всегда применяться одна и та же интерпретация, одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;

• обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;

• обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Каждая сущность может обладать любым количеством связей с другими сущностями модели.

Связь (*Relationship*) – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь – это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, и наоборот.

Атрибут (*Attribute*) – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, предметов и т. д.).

Экземпляр атрибута – это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. На диаграмме «сущность-связь» атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

3.2. Метод Чена

Символы ERD, соответствующие сущностям и отношениям, ис-

пользуемые при построении диаграмм «сущность-связь» в нотации Чена, приведены на рисунке 18.

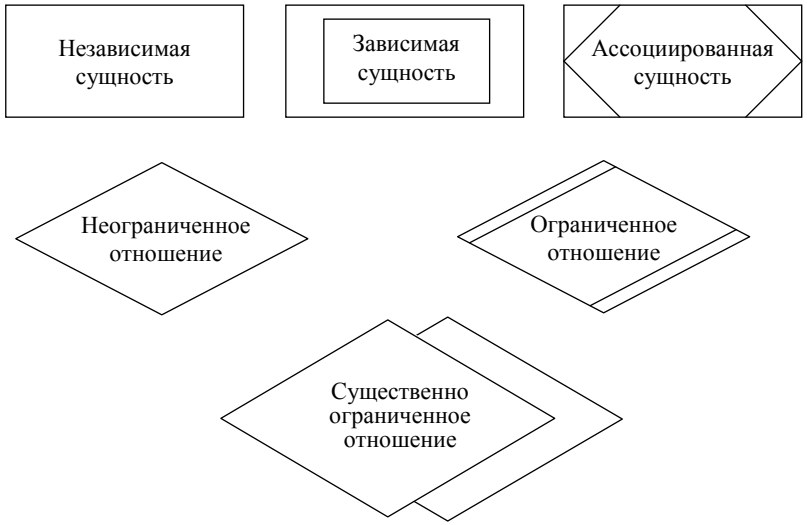

Рисунок 18 – **Символы диаграмм «сущность-связь» в нотации Чена**

Независимая сущность представляет независимые данные, которые всегда присутствуют в системе. При этом отношения с другими сущностями могут как существовать, так и отсутствовать. В свою очередь *зависимая сущность* представляет данные, зависящие от других сущностей в системе. Поэтому она должна всегда иметь отношения с другими сущностями. *Ассоциированная сущность* представляет данные, которые ассоциируются с отношениями между двумя и более сущностями.

Неограниченное (обязательное) отношение представляет собой безусловное отношение, т. е. отношение, которое всегда существует до тех пор, пока существуют относящиеся к делу сущности. *Ограниченное (необязательное) отношение* представляет собой условное отношение между сущностями. *Существенно ограниченное отношение* используется, когда соответствующие сущности взаимно зависимы в системе.

Для идентификации требований, в соответствии с которыми сущности вовлекаются в отношения, используются *связи.* Каждая связь соединяет сущность и отношение и может быть направлена только от отношения к сущности.

Значение связи характеризует ее тип и, как правило, выбирается из следующего множества:

{«0 или 1», «0 или более», «1», «1 или более», «p:q» (диапазон)}.

Пара значений связей, принадлежащих одному и тому же отношению, определяет тип этого отношения. Практика показала, что для большинства приложений достаточно использовать следующие типы отношений:

- *отношения 1:1 (один-к-одному)*, используемые, как правило, на верхних уровнях иерархии модели данных, а на нижних уровнях встречаются сравнительно редко;
- *отношения 1:n (один-ко-многим)*, являющиеся наиболее часто используемыми;
- *отношения n:m (многие-ко-многим)*, которые обычно используются на ранних этапах проектирования с целью прояснения ситуации; в дальнейшем каждое из таких отношений должно быть преобразовано в комбинацию отношений первых двух типов (возможно, с добавлением вспомогательных сущностей и введением новых отношений).

На рисунке 19 приведена диаграмма «сущность-связь», демонстрирующая отношения между сущностями из примера подраздела 2.4.

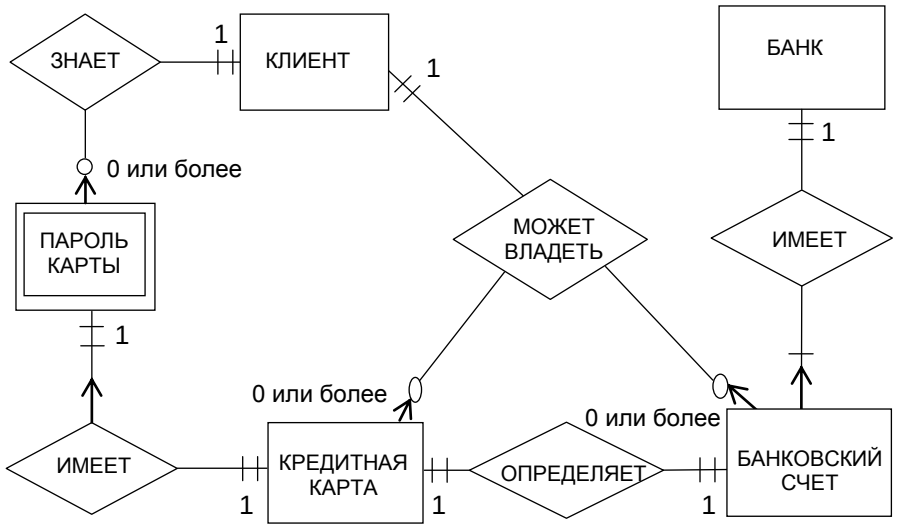

Рисунок 19 – **Пример диаграммы «сущность-связь» в нотации Чена**

Согласно этой диаграмме каждый БАНК ИМЕЕТ один или более

БАНКОВСКИХ СЧЕТОВ. Кроме того, каждый КЛИЕНТ МОЖЕТ ВЛАДЕТЬ (одновременно) одной или более КРЕДИТНОЙ КАРТОЙ и одним или более БАНКОВСКИМ СЧЕТОМ, каждый из которых ОПРЕДЕЛЯЕТ в точности одну КРЕДИТНУЮ КАРТУ (отметим, что у клиента может не быть ни счета, ни кредитной карты). Каждая КРЕДИТНАЯ КАРТА ИМЕЕТ ровно один зависимый от нее ПАРОЛЬ КАРТЫ, а каждый КЛИЕНТ ЗНАЕТ (но может и забыть) ПАРОЛЬ КАРТЫ.

3.3. Метод Баркера

Одной из наиболее распространенных разновидностей нотации ERD является нотация, предложенная Ричардом Баркером, автором методов, используемых в технологии создания программного обеспечения фирмы Oracle.

Метод Баркера поясним на примере моделирования данных компании по торговле автомобилями. Исходными данными для построения ERD являются результаты интервью, проведенного с персоналом компании, выдержки из которого приведены ниже.

Главный менеджер. Одна из его основных обязанностей – содержание автомобильного имущества. Он должен знать, сколько заплачено за машины и каковы накладные расходы. Обладая этой информацией, он может установить нижнюю цену, за которую мог бы продать данный экземпляр. Кроме того, он несет ответственность за продавцов и ему нужно знать, кто что продает и сколько машин продал каждый из них.

Продавец. Ему нужно знать, какую цену запрашивать и какова нижняя цена, за которую можно совершить сделку. Кроме того, ему нужна основная информация о машинах: год выпуска, марка, модель и т. п.

Администратор. Его задача сводится к составлению контрактов, поэтому ему нужна информация о покупателе, автомобиле и продавце, поскольку именно контракты приносят продавцам вознаграждения за продажи.

Первый шаг моделирования – извлечение информации из интервью и выделение сущностей.

Обращаясь к приведенным выше выдержкам из интервью, можно увидеть, что сущности, которые могут быть идентифицированы главным менеджером, – это автомобили и продавцы. Продавцу важны автомобили и связанные с их продажей данные. Для администратора

важны покупатели, автомобили, продавцы и контракты.

Исходя из этого выделяются четыре сущности:

- автомобиль;
- продавец;
- покупатель;
- контракт.

Второй шаг моделирования – идентификация связей.

Определение связи в методе Баркера несколько отличается от определения, данного Ченом. Связь – это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой *родительской сущностью*, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой *сущностью-потомком*, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности-родителя.

Связи может даваться имя, выраженное в форме глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя.

Например, связь продавца с контрактом может быть выражена следующим образом:

- продавец может получить вознаграждение за один контракт или более;
- контракт должен быть инициирован ровно одним продавцом.

Степень и обязательность связи отражается графически (рисунок 20).

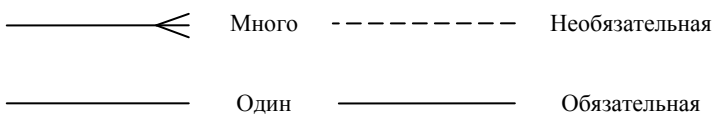

Рисунок 20 – **Степень и обязательность связи**

Изобразим графически предложения, описывающие связь продавца с контрактом (рисунок 21).

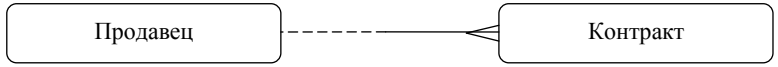

Рисунок 21 – **Связь продавца с контрактом**

Аналогичным образом изобразим графически предложения, описывающие связь покупателя с контрактом (рисунок 22) и связь автомобиля с контрактом (рисунок 23).

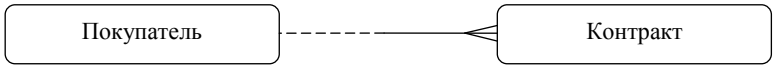

Рисунок 22 – **Связь покупателя с контрактом**

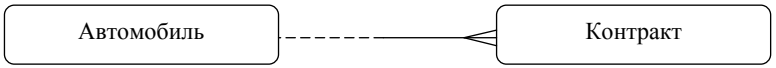

Рисунок 23 – **Связь автомобиля с контрактом**

Третий шаг моделирования – идентификация атрибутов.

Атрибут может быть либо обязательным, либо необязательным (рисунок 24).

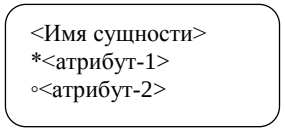

Рисунок 24 – **Обязательный (помечен звездочкой) и необязательный (помечен кружком) атрибуты**

Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор – это атрибут или совокупность атрибутов и (или) связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя (рисунок 25).

а б

Рисунок 25 – **Виды идентификации:**

а – полная идентификация;

б – идентификация посредством другой сущности

Каждый атрибут идентифицируется уникальным именем, выраженным существительным, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком «#».

Каждая сущность должна обладать хотя бы одним возможным ключом. *Возможный ключ сущности* – это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные – как альтернативные ключи.

С учетом имеющейся информации дополним построенную ранее диаграмму (рисунок 26).

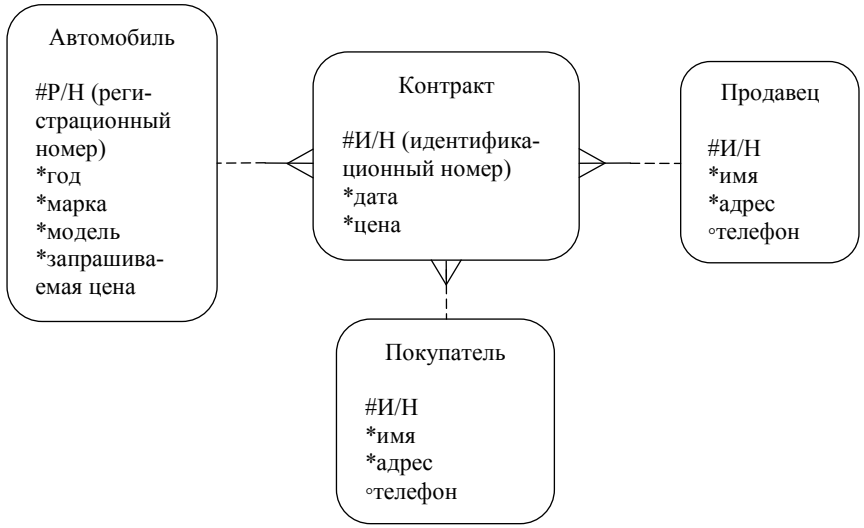

Рисунок 26 – **Диаграмма «сущность-связь» с атрибутами**

3.4. Метод IDEF1

Метод IDEF1 удобен для моделирования реляционных баз данных. В настоящее время на основе совершенствования метода IDEF1 создана его новая версия – метод IDEF1X, разработанный с учетом та-

кого требования, как простота для изучения.

В методе IDEF1X выделяются два типа сущностей: независимая и зависимая. Сущность является *независимой от идентификаторов* или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется *зависимой от идентификаторов* или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Условные обозначения сущностей метода IDEF1X приведены на рисунке 27.

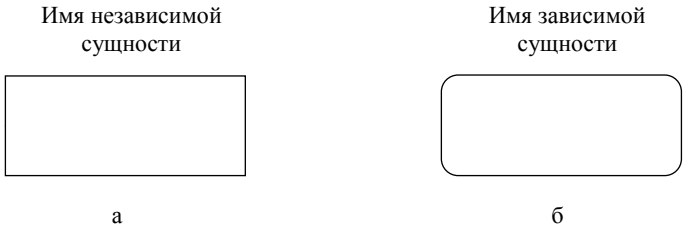

Рисунок 27 – **Условные обозначения независимой (а) и зависимой (б) от идентификатора сущности**

Связь между сущностями на IDEF1X-диаграммах изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка, причем связь между независимой и зависимой сущностями изображается сплошной линией (рисунок 28).

Рисунок 28 – **Пример графического изображения связи между сущностями**

Внутри условных обозначений сущностей на IDEF1X-диаграммах может быть горизонтальная линия, над которой указывается первичный ключ сущности. Как и в любой другой диаграмме «сущность-связь» первичный ключ предназначен для уникальной идентификации экземпляра сущности. Под горизонтальной линией в этом случае указываются все остальные атрибуты данной сущности.

Сущности могут иметь также внешние ключи (Foreign Key), которые автоматически образуются в дочерней сущности при указании соответствующей связи (рисунок 29).

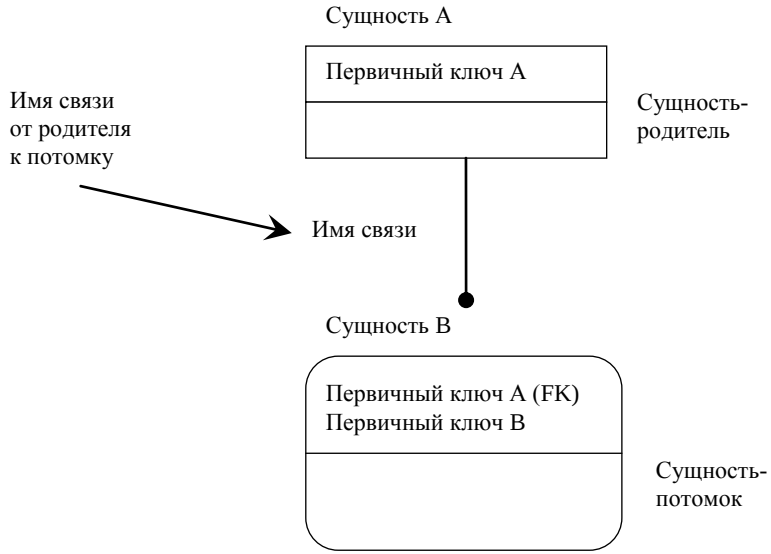

Рисунок 29 – **Идентифицирующая связь**

Связь между сущностями может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которые могут существовать для каждого экземпляра сущности-родителя).

В IDEF1X могут быть выражены следующие мощности связей:

• Каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка (рисунок 30).

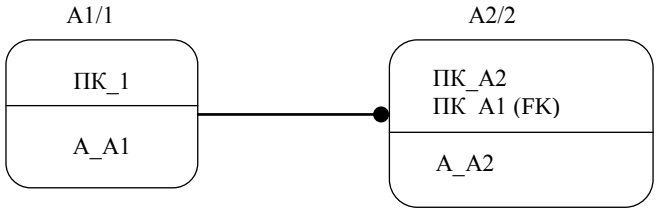

Рисунок 30 – **Графическое изображение связи между сущностями, когда одному экземпляру родительской сущности соответствуют**

0, 1 или много экземпляров дочерней сущности

• Каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка (рисунок 31).

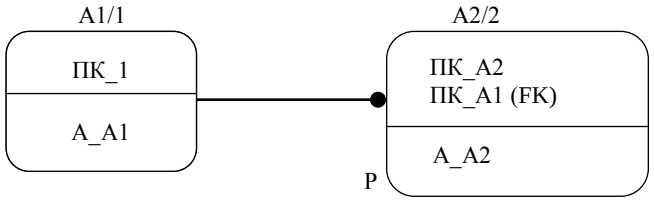

Рисунок 31 – **Графическое изображение связи между сущностями, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности**

• Каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка (рисунок 32).

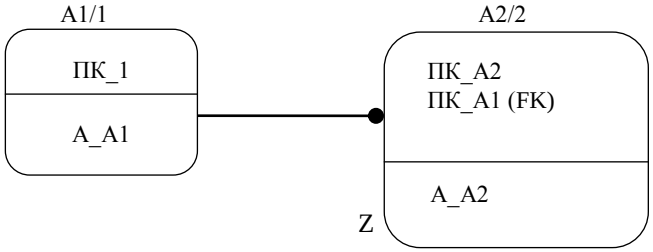

Рисунок 32 – **Графическое изображение связи между сущностями, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляров дочерней сущности**

• Каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка (рисунок 33).

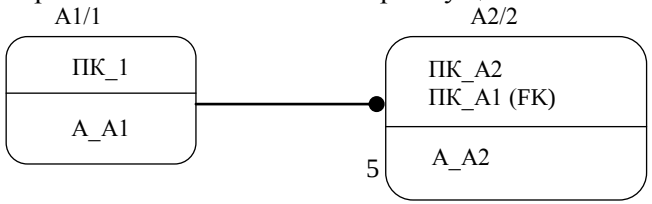

Рисунок 33 – **Графическое изображение связи между сущностями, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности**

Связь между независимыми сущностями на IDEF1X-диаграммах (неидентифицирующая связь) изображается пунктирной линией.

В качестве примера IDEF1X-диаграммы на рисунке 34 приведена логическая модель базы данных, содержащей информацию о студентах факультета вуза.

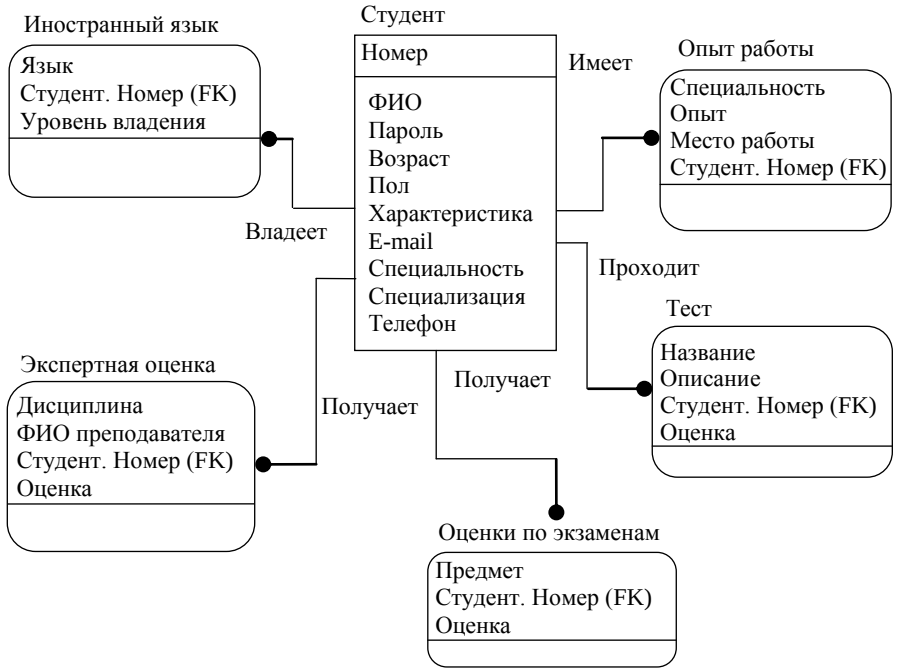

Рисунок 34 – **Пример IDEF1X-диаграммы**

Метод IDEF1X позволяет отобразить более сложные зависимости между сущностями, например, отношения категоризации – отношения между двумя и более сущностями, в которых каждый экземпляр одной сущности, называемой общей, связан в точности с одним экземпляром сущности, называемой сущностью-категорией.

На рисунке 35 приведен пример изображения иерархии категорий на IDEF1X-диаграммах.

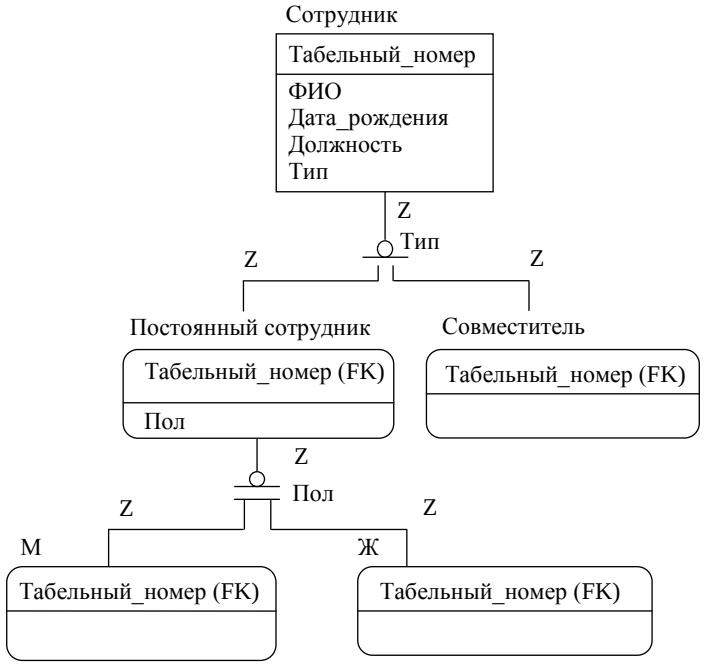

Рисунок 35 – **Пример иерархии категорий**

Категория выделяется из общей сущности по определенному признаку, при этом различают полную и неполную категоризацию. В примере на рисунке 34 категоризация по признаку «Тип» неполная, так как кроме постоянных сотрудников и совместителей могут быть другие сотрудники (например, консультанты), а категоризация по признаку «Пол» полная.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Структурный анализ систем: IDEF-технологии / авт.-сост. : С. В. Черемных, И. О. Семенов, В. С. Ручкин. – М. : Финансы и статистика, 2001. – 208 с.

Федотова, Д. Э. CASE-технологии : практикум / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – М. : Горячая линия-Телеком, 2005. –

160 с.

Калашян, А. Н. Структурные модели бизнеса: DFD-технологии / А. Н. Калашян, Г. Н. Калянов. – М. : Финансы и статистика, 2003. – 256 с.

Марко, Д. А. Методология структурного анализа и проектирования / Д. А. Марко, К. Мак Гоуэн. – М. : МетаТехнология, 1993. – 211 с.

Вендров, А. М. CASE-технологии. Современные методы и средства проектирования информационных систем / А. М. Вендров. – М. : Финансы и статистика, 1998. – 151 с.

Вендров, А. М. Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. – М. : Финансы и статистика, 2000. – 231 с.

СОДЕРЖАНИЕ

Пояснительная записка 3
1. Метод структурного анализа и проектирования SADT 5
1.1. Общие сведения 5
1.2. Основные элементы IDEF0-диаграмм 5
1.3. Построение иерархии диаграмм 10
1.4. Пример построения IDEF0-диаграмм деятельности организации 14
1.4.1. Постановка задачи 14
1.4.2. Построение контекстной диаграммы 14
1.4.3. Построение диаграммы декомпозиции первого уровня 15
1.4.4. Построение диаграммы декомпозиции функционального блока «Сборка компьютеров» 17
1.4.5. Результаты анализа 17
2. Диаграммы потоков данных 19
2.1. Общие сведения 19
2.2. Основные элементы DFD-диаграмм 19
2.3. Построение иерархии DFD-диаграмм 21
2.4. Пример построения DFD-диаграмм 23
2.5. Словарь данных 27
2.6. Спецификации процессов DFD-диаграмм 28
3. Диаграммы «сущность-связь» 30
3.1. Общие сведения 30
3.2. Метод Чена 31
3.3. Метод Баркера 34
3.4. Метод IDEF1 37

Список рекомендуемой литературы 42

Учебное издание

МЕТОДЫ
СТРУКТУРНОГО АНАЛИЗА
ИНФОРМАЦИОННЫХ СИСТЕМ

Пособие
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Автор-составитель
Семенюта Андрей Николаевич

Редактор М. П. Герасенко
Технический редактор И. А. Козлова
Компьютерная верстка Н. Н. Короедова

Подписано в печать 17.03.11. Бумага типографская № 1.
Формат 60 × 84 $^{1}/_{16}$. Гарнитура Таймс. Ризография.
Усл. печ. л. 2,56. Уч.-изд. л. 2,60. Тираж 70 экз.
Заказ №

Учреждение образования
«Белорусский торгово-экономический университет
потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.
ЛИ № 02330/0494302 от 04.03.2009 г.

Отпечатано в учреждении образования
«Белорусский торгово-экономический университет
потребительской кооперации».

246029, г. Гомель, просп. Октября, 50.
БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»

Кафедра информационно-вычислительных систем

МЕТОДЫ СТРУКТУРНОГО АНАЛИЗА ИНФОРМАЦИОННЫХ СИСТЕМ

Пособие
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Гомель 2011